BASN Chip Memory Documentation

By: Jim Boley

Tape out date: November 2013

Memory Overview:

Because the BASN chip operates in subthreshold, the bitcell chosen for this application was the 8T cell (Figure 1). This cell was chosen because it uses a two transistor read buffer which allows the cell to be read without disturbing the data being stored. This in turn allows the 6T cell to be fully optimized for writeability. Because the IBM 130 technology is very N-strong, the NMOS passgate transistor can very easily flip the cross coupled inverter pair which makes for high write margins and eliminates the need for write assist methods. Because read static noise margin is not an issue with the 8T cell, only write and hold margins were considered. The lengths and widths of each transistor in the bitcell were optimized using TASE. The sizing is shown below:

|  |  |  |
| --- | --- | --- |
|  | Width | Length |
| PU | 400n | 220n |
| PD | 280n | 150n |
| PG | 450n | 220n |
| Read-buf | 360n | 150n |

One problem faced by subthreshold memories is that bitline leakage through unaccessed cells can result in the RBL being pulled low incorrectly during a read ‘1’ (this occurs when the total leakage of the unaccessed rows is greater than the leakage through the PMOS precharge device). To solve this problem, the “zero leakage read-buffer” method described by [Verma and Chandrakasan](http://www-mtl.mit.edu/researchgroups/icsystems/pubs/journals/2008_verma_jssc_jan.pdf) was employed. This method charges the footer of unaccessed rows up to VDD which virtually eliminates leakage current in unaccessed rows. Because the footer of an accessed row must also sink the read current of the entire row, a charge pump is needed to push the gate of this NMOS transistor to VDD\*2, giving it more drive strength. Monte carlo simulations were run to ensure that the read operation would not fail due to local or global variation.

Due to time constraints, the architecture of the memory was kept simple. Both the instruction and data memory consisted of four blocks. Each block contained a bitcell array along with row and column periphery and output sense amps (Figure 2). The row and block decoders use an enable signal to ensure that word lines are only pulsed when the clock and enable signals are high. This prevents the decoders from glitching and accidently writing a row. The precharge device is setup so that if the clock is low, the bitlines are precharging. This means that read data must be latched before the falling edge of the clock. In order to prevent read data upsets, the internal clock was delayed 1 us with respect to the external clock. This means that the external registers (which are negative edge triggered) have over 1 us to latch the read data before the precharge cycle begins. All inputs to the memory are held in positive edge triggered flip flops. Because these flops have a clock to Q delay, the delayed clock ensures that the data being inputted to the memory becomes valid before the rising edge of the delayed clock (See figure 3). The delayed clock method works well with this design because the memory is expected to run only 200 Khz.

Instruction Memory:

4 blocks \* 384 bytes/block = 1536 total bytes or 1024 total instructions

This memory also contains 88 twelve bit instructions stored in ROM cells. The ROM cells were created by using the 2T read buffer and tying QB to either VDD or VSS. The cells were created in layout by deleting the 6T portion of the cell, leaving only the two read transistors. This made it easy to replace RAM cells with ROM cells. Because the instruction set was 1056 bits long, a script was used to read in the bit stream and tile the appropriate ROM cells.

Data Memory:

4 blocks \* 1024 bytes/block = 4096 total bytes

Because this chip is extremely low power, it is necessary to sleep unused memory blocks to reduce leakage current. This was done by placing a 24 micron wide footer onto each block. In order to sink the necessary on current, the gate of the footer was overdriven to 1.2 volts by a level converter. Each block was controlled by a separate sleep signal, allowing each block to be slept independently. In order to prevent LVS errors, the bulks of all NMOS transistors were tied to the global VSS as opposed to the virtual VSS.

*Figure 1: The 8T Bitcell*

****

*Figure 2: High Level Block Diagram of the SRAM*

ADR<0:4>

Global WL<0:31>

CLK\_EN

ADR<5:6>

ADR<5:6>

CLK\_EN

ENABLE

Data\_sel\_blk<0:3>

Blk\_sel\_clk\_en<0:3>

384 Bytes

Blk\_sel\_clk\_en<0>

READWB

READWB

Global WL<0:31>

RWL<0:31>

WWL<0:31>

READWB

Data\_sel\_blk<0>

DIN<0:95>

VDD

Blk\_sel\_clk\_en<0>

RBL<0:95>

WBL/B<0:95>

RBL<0:95>

DOUT<0:95>

Output Mux

**x4**

DOUT\_BLK0<0:95>

DOUT\_BLK1<0:95>

DOUT\_BLK2<0:95>

DOUT\_BLK3<0:95>

DOUT <0:95>

CLOCK

CLK\_EN

CLK\_Delay

CLK\_Delay

Clock (200 KHz)

IDLE

WRITE

READ

ENABLEB

READWB

CLK\_EN (RWL)

CLK\_EN\_BAR (LATCH\_CLK)

CLKB\_EN

CLKB\_EN ***&&*** !READWB (WWL)

CLK\_EN ***NAND*** READWB (OUT\_FF\_CLK)

Hold

Transparent

Hold

Transparent

Hold

Latching edge

Clock (200 KHz)

Read ADR 0

READ ADR 1

ENABLEB

READWB

ADR 0 Data valid

ADR<0>

ADR 0 Data valid

Idle

ADR 0 Data latched externally (set up time= 2.5us, hold time = 2.5us)